



Angular Fundamentals

Module - Routing

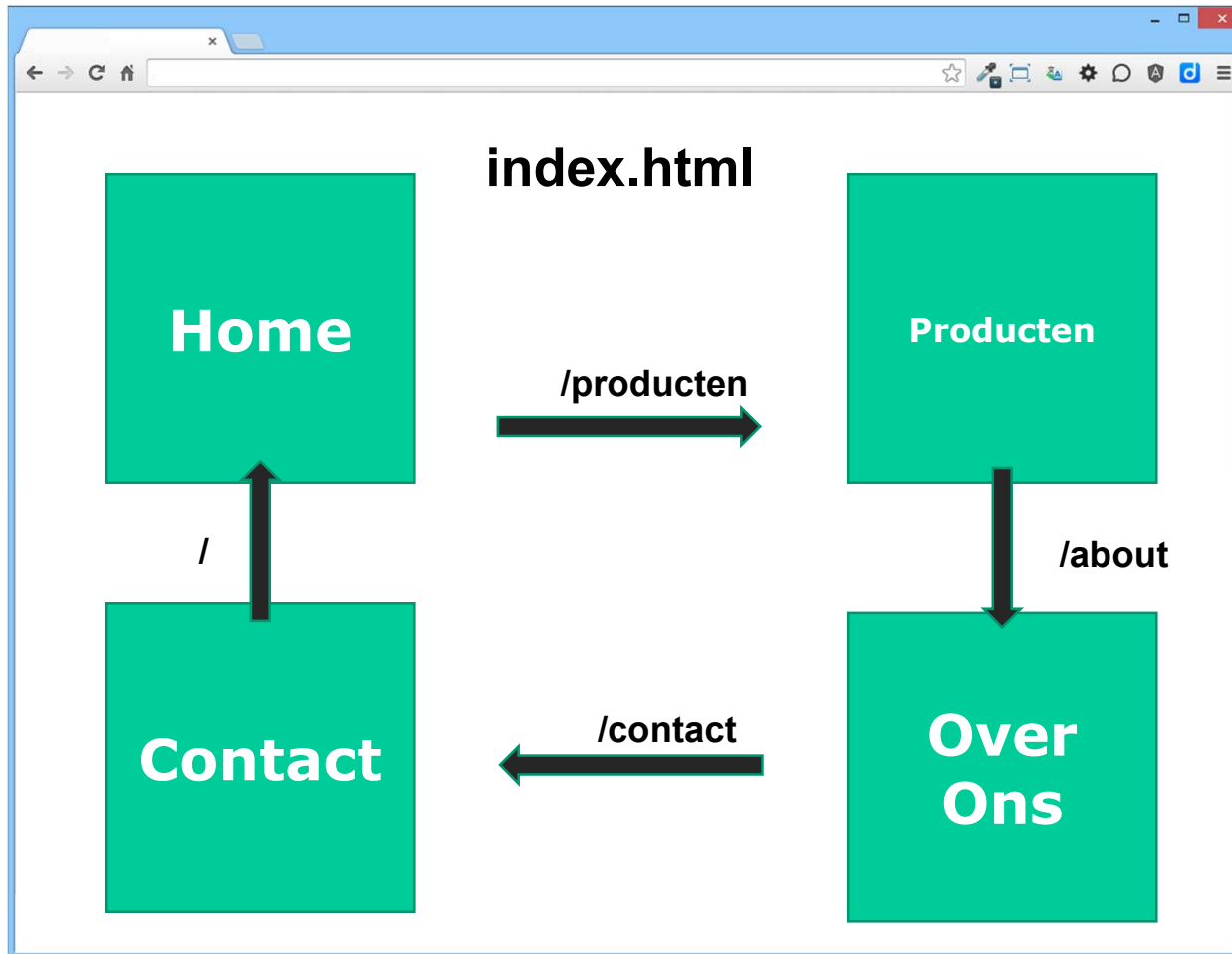


Peter Kassenaar
info@kassenaar.com



Hoofdstuk 8
p. 216 en verder

Routing architecture and goal



- Make use of SPA principle
- Making deep links possible

Angular 1: ng-route, of ui-router

1. `<script src="js/vendor/angular/angular-route.min.js"></script>`

2. `<div ng-view></div>`

3. `var app = angular.module('myApp', ['ngRoute']);`

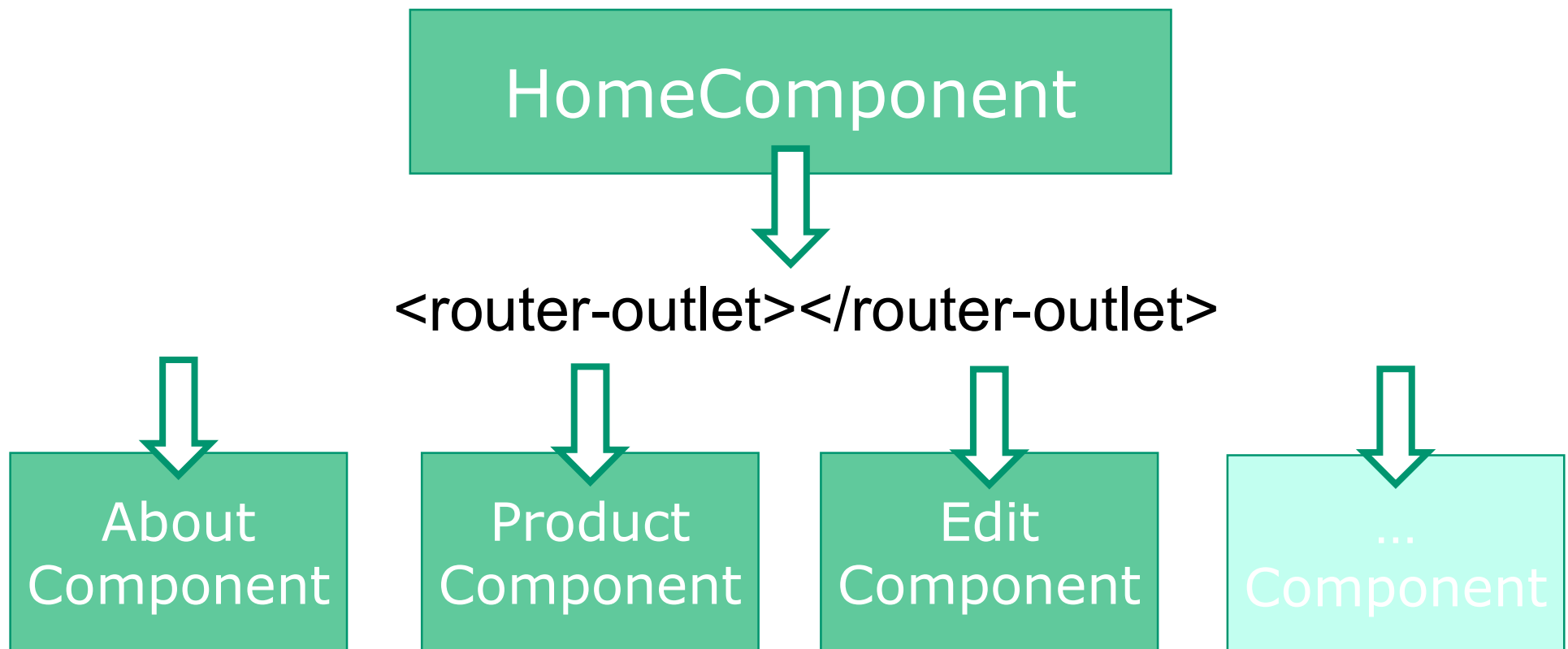
Daarna `$routeProvider` configureren (of `$stateProvider` bij ui-router)

Angular 2: Component Router

- Niet beschikbaar voor AngularJS 1.4+
- Niet veel gebruikt: ui-router

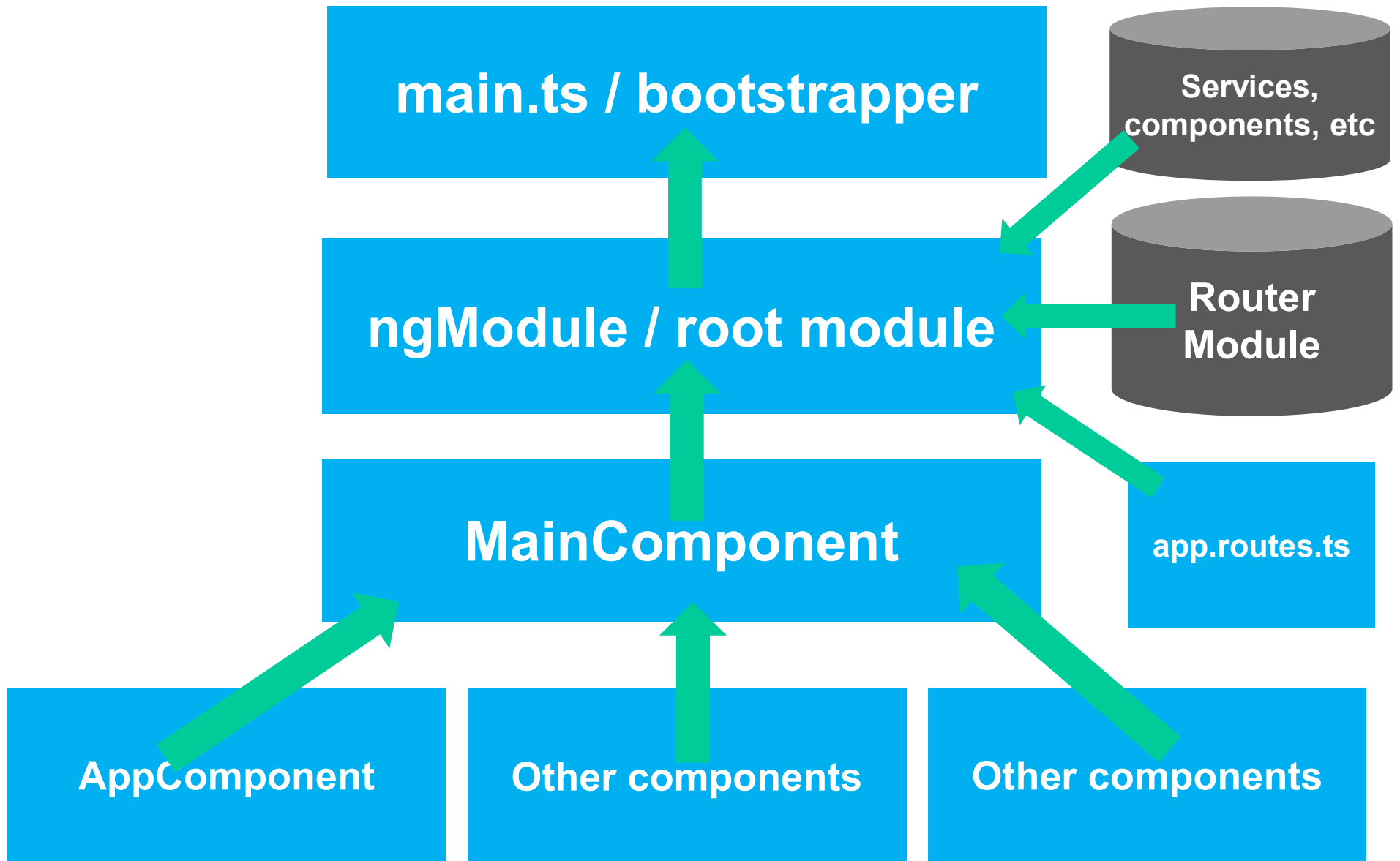
Routing – every route is a Component

- HomeComponent (or: RootComponent, whatever) with main menu
- Components are injected in `<router-outlet></router-outlet>`



Routing met Angular CLI

- Standaard: géén routing in CLI-project
- Routing vanaf het begin toevoegen?
 - `ng new myProject --routing`
 - OF: vraag in CLI-command prompt met `Yes` beantwoorden
- Dit maakt `app.routing.module` in het project
- (iets) anders van opbouw dan we hier presenteren



Stappenplan routing

1. Base Href toevoegen in header van index.html (!)

<base href="/">

- Er *kunnen* meerdere routes per module zijn. Elke component kan zijn eigen `ChildRoutes` definiëren – volgt later.
- Angular-CLI doet dit automatisch voor je.

2. Routes toevoegen. Convention: `app.routes.ts`.

```
// app.routes.ts
import {Routes} from '@angular/router';
import {AppComponent} from './app.component';
import {CityAddComponent} from './city.add.component';

export const AppRoutes: Routes = [
  {path: '', component: AppComponent},
  {path: 'home', component: AppComponent},
  {path: 'add', component: CityAddComponent}
];
```

Er zijn meerdere opties en notatiewijzen om routes te declareren

3. Routes beschikbaar maken in Module

- Import RouterModule in applicatie
- Import ./app.routes in applicatie

```
...  
// Router  
import {RouterModule} from '@angular/router';  
import {AppRoutes} from './app.routes';
```

Import Router-
onderdelen

```
// Components  
import {MainComponent} from './MainComponent';
```

Nieuw!
MainComponent
gaan we nog maken

```
...  
@NgModule({  
  imports : [  
    BrowserModule, HttpClientModule,  
    RouterModule.forRoot(AppRoutes)  
  ],  
  declarations: [  
    MainComponent,  
    AppComponent,  
    CityAddComponent  
  ],  
  bootstrap : [MainComponent]  
})  
export class AppModule {  
}
```

Configure
RouterModule.forRoot()

MainComponent wordt nu
gebootstrapt

4. MainComponent met Routing maken

- Nieuwe component met hoofdmenu en `<router-outlet>`

```
import {Component, OnInit} from '@angular/core';
```

```
@Component({
  selector: 'main-component',
  template: `
    <h1>Pick your favorite city</h1>
    <!-- Static 'main menu'. Always visible-->
    <!-- Add routerLink directive. Angular replaces this with correct <a href="..."> -->
    <a routerLink="/home" class="btn btn-primary">List of cities</a>
    <a routerLink="/add" class="btn btn-primary">Add City</a>
    <hr>
    <!-- Dynamically inject views here -->
    <router-outlet></router-outlet>
    <!-- Static footer here. Always visible-->
  `
})
export class MainComponent implements OnInit {
  constructor() { }
  ngOnInit() { }
}
```

"Hoofdmenu". Let op routerLink

<router-outlet>

Lege Component

5. Eventueel: index.html aanpassen

- Eventueel selector in index.html aanpassen
- Als `MainComponent` een andere selector heeft

```
<div class="container">  
  <main-component>  
    Loading...  
  </main-component>  
</div>
```

6. Nieuwe component(en) maken en importeren

Elke component is een route

```
// city.add.component.ts
import { Component } from 'angular2/core';
```

```
@Component({
  selector: 'add-city',
  template: `<div>Add City</div>`
})
export class CityAddComponent {
  ...
}
```

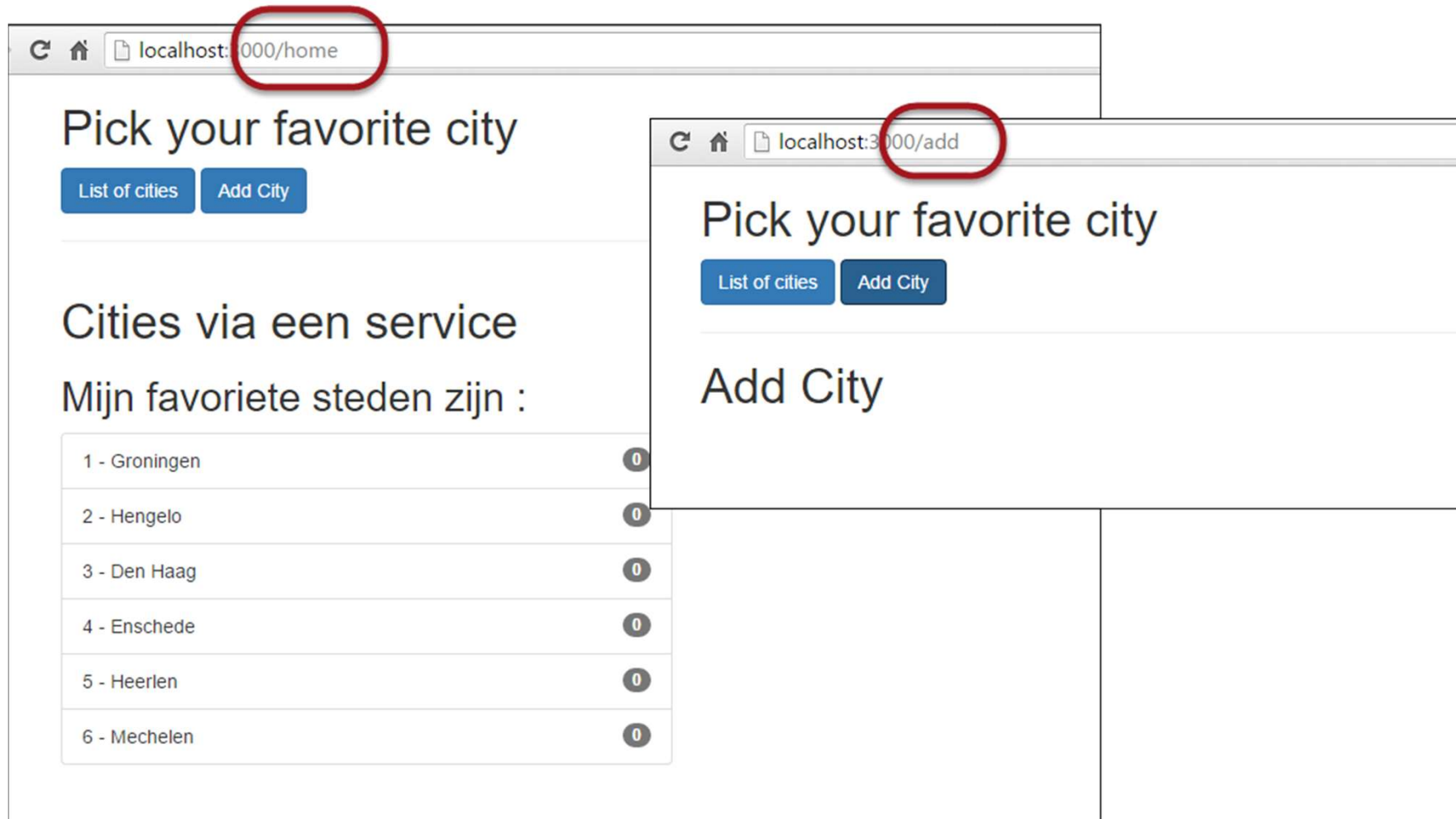
```
// city.edit.component.ts
import { Component } from 'angular2/core';
```

```
@Component({
  selector: 'edit-city',
  template: `<h1>Edit City</h1>`
})
export class CityEditComponent {
  ...
}
```

```
// city.detail.component.ts
import { Component } from 'angular2/core';
```

```
@Component({
  selector: 'detail-city',
  template: `<h1>Detail City</h1> ...`
})
export class CityDetailComponent {
  ...
}
```

7. Run the application



Catch-all routes

```
6 export const AppRoutes: Routes = [  
7   {path: '', component: AppComponent},  
8   {path: 'home', component: AppComponent},  
9   {path: 'add', component: CityAddComponent},  
10  {  
11    // catch all route  
12    path: '**',  
13    redirectTo: 'home'  
14  },  
15 ];
```

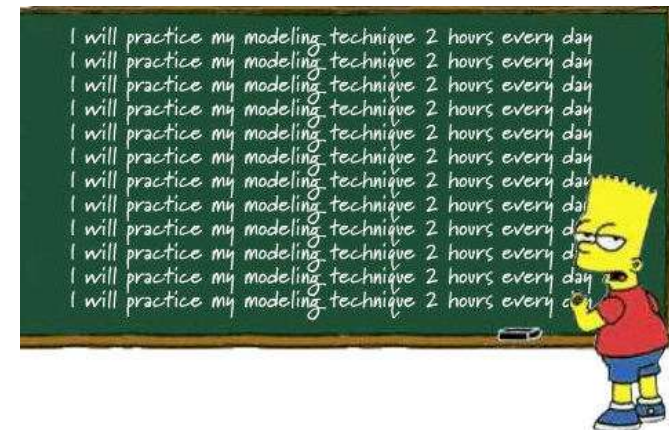
Gebruik `**` voor een catch-all route:

- Component opgeven (=route blijft zichtbaar in URL-balk)
- `redirectTo`: opgeven (=nieuwe route staat in URL-balk)

Checkpoint

- Routes worden op module-level ingesteld (Angular 1: app-level).
- Volg het stappenplan. Denk aan injecteren van RouterModule, `app.routes.ts` en `<base href="/">` in de HTML
- Voorbeeld: `/400-routing`
- Voeg een nieuwe component toe aan het routing-voorbeeld en zorg dat naar deze component kan worden gerouteerd
- Oefening 7a) en 7b) (=nieuwe app maken, inclusief `--routing`)

Oefening....





Routeparameters

Master-Detail views en –applications

Dynamische routes maken

Doel: Enkele detailpagina voor klanten, producten, diensten, etc.

Leesbare routes als: `/cities/5`, of `products/philips/broodrooster`, enzovoort

Werkwijze:

1. Aanpassen `routing table` in de applicatie.
2. Gebruik `route:ActivatedRoute` in de detail component
3. Schrijf hyperlinks als `<a [routerLink]=...>` met parameter

1. app.routes.ts aanpassen

```
// app.routes.ts
import {Routes} from '@angular/router';
import {AppComponent} from "../app.component";
import {CityAddComponent} from "../city.add.component";
import {CityDetailComponent} from "../city.detail.component";

export const AppRoutes: Routes = [
  {path: '', component: AppComponent},
  {path: 'home', component: AppComponent},
  {path: 'add', component: CityAddComponent},
  {path: 'detail/:id', component: CityDetailComponent}
];
```



2. Detail Component maken

```
// city.detail.component.ts
...
// import {RouteParams} from "@angular/router"; // OLD way
import {ActivatedRoute} from '@angular/router';

@Component({
  selector: 'city-detail',
  template: `

# City Detail</h1> <h2>Details voor city: {{ id }}</h2> ` }) export class CityDetailComponent implements OnInit, OnDestroy { id: string; currentCity: City; constructor(private route: ActivatedRoute) {} ngOnInit() { this.route.params .subscribe((id: any) => { this.id = id; }); } }


```



ActivatedRoute

2a. DetailComponent - variants

Using router snapshots

```
// OR:  
// Work via Router-snapshot:  
// Sometimes we're not interested in future changes of a route parameter.  
// All we need the id and once we have it, we can provide the data we want to provide.  
// In this case, an Observable can bit a bit of an overkill.  
// A *snapshot* is simply a snapshot representation of the activated route.  
this.id = this.route.snapshot.params['id'];  
this.name = this.route.snapshot.params['name'];
```

2b. DetailComponent - variants

```
ngOnInit() {  
  // NEW:  
  this.sub = this.route.params  
    .subscribe((params: any) => {  
    this.id = params['id'];  
    this.name = params['name'];  
  });  
}
```

```
ngOnDestroy() {  
  // If subscribed, we must unsubscribe before Angular destroys the component.  
  // Failure to do so could create a memory leak.  
  this.sub.unsubscribe();  
}
```



.unsubscribe()

3. Detail component toevoegen aan Module

```
// app.module.ts
...
// Components
import {CityDetailComponent} from './city.detail.component';


@NgModule({
  imports      : [
    ...
  ],
  declarations: [
    ...
    CityDetailComponent
  ],
  providers    : [CityService],
  bootstrap    : [MainComponent]
})
export class AppModule {
}
```



Component

3. App Component ('Master View') aanpassen

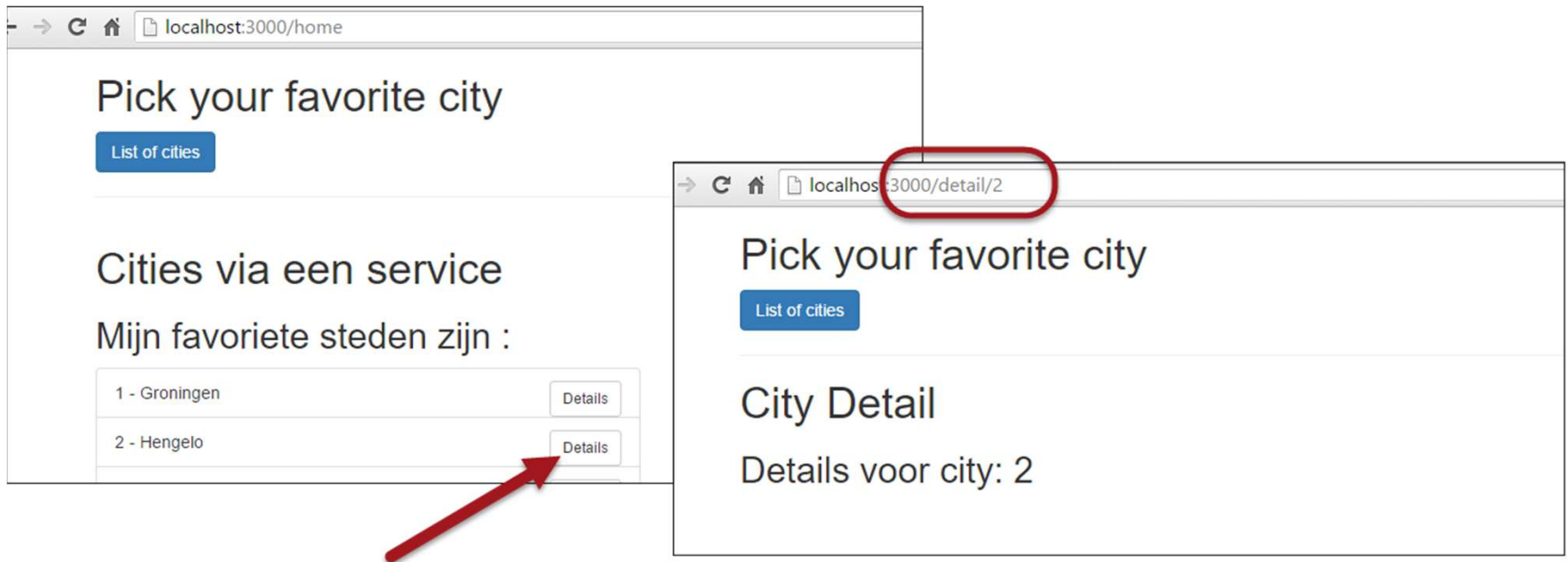
```
<li *ngFor="let city of cities" class="list-group-item">  
  <a [routerLink]="['/detail', city.id]">  
    {{ city.id }} - {{ city.name }}  
  </a>  
</li>
```



Let er op dat `[routerLink]` nu dynamisch moet worden gevuld en dus binnen `[...]` moet staan voor attribute binding

Meegeven van parameters

- Let op meegeven van *array van parameters* aan `[routerLink]`
- Parameters worden gematched op positie. Niet op naam.
- Optioneel : service uitbreiden om specifiek product/item te retourneren





Optionele parameters

Optionele routeparameters meegeven op de querystring

Optionele parameters : [queryParams]

In de HTML

```
<a [routerLink]="['/detail', city.id, city.name]"
  [queryParams]="{province:city.province, population:180000}">
  {{ city.id }} - {{ city.name }}
</a>
```

In de class

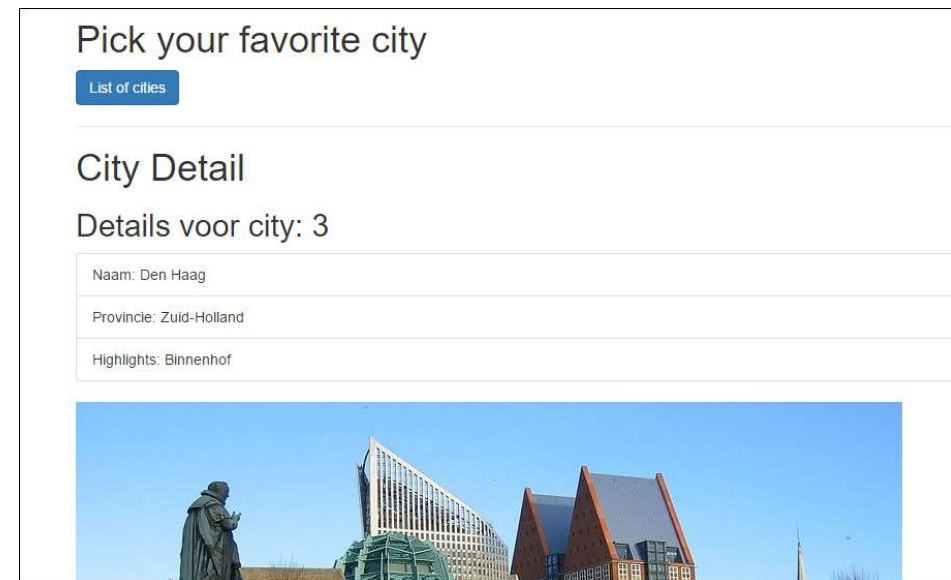
```
this.route.queryParams.subscribe((params: any) => {
  this.province = params.province;
})
```

Vervolg – details via Service

- Uncomment de regels die te maken hebben met cityService:

// NEW, with fetching details via Service:

```
this.sub = this.route.params
  .map(params => params['id'])
  .switchMap(id => this.cityService.getCity(id))
  .subscribe((city) => {
    this.currentCity = <City>city[0];
  });
```



In city.service.ts:

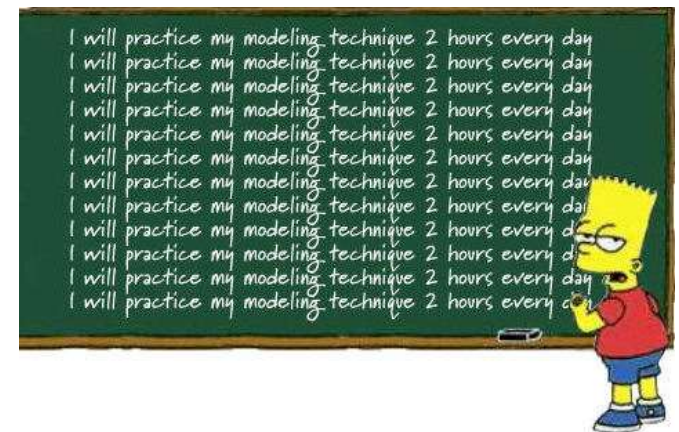
- Bijvoorbeeld (kan beter, maar het werkt wel):

```
// retourneer een city, op basis van ID
getCity(id: string): Observable<City[]> {
    return this.http.get('app/cities.json')
        .pipe(
            map(cities => cities.filter((city: City) => {
                return city.id === parseInt(id);
            }))
        )
}
```

Checkpoint

- RouteParameters worden met `:parameterName` ingesteld in `app.routes.ts`.
- Denk aan injection van `ActivatedRoute` in de component.
- Hierin is een property `.params` aanwezig met de meegegeven parameters.
- `<a [routerLink]="...">` gaan uitbreiden
- Voorbeeld: `\401-router-parameter`
- Oefening: 7c)

Oefening....



More on routing

- Router Guards – delen van je routes beveiligen
- Child Routes
- Named Router Outlets
 - <http://onehungrymind.com/named-router-outlets-in-angular-2/>
- Router resolvers
 - <https://blog.thoughttram.io/angular/2016/10/10/resolving-route-data-in-angular-2.html>
- Lazy Loading – Applicatie opdelen in Modules en laden *on demand*
 - <https://angular.io/guide/router#lazy-loading-route-configuration>



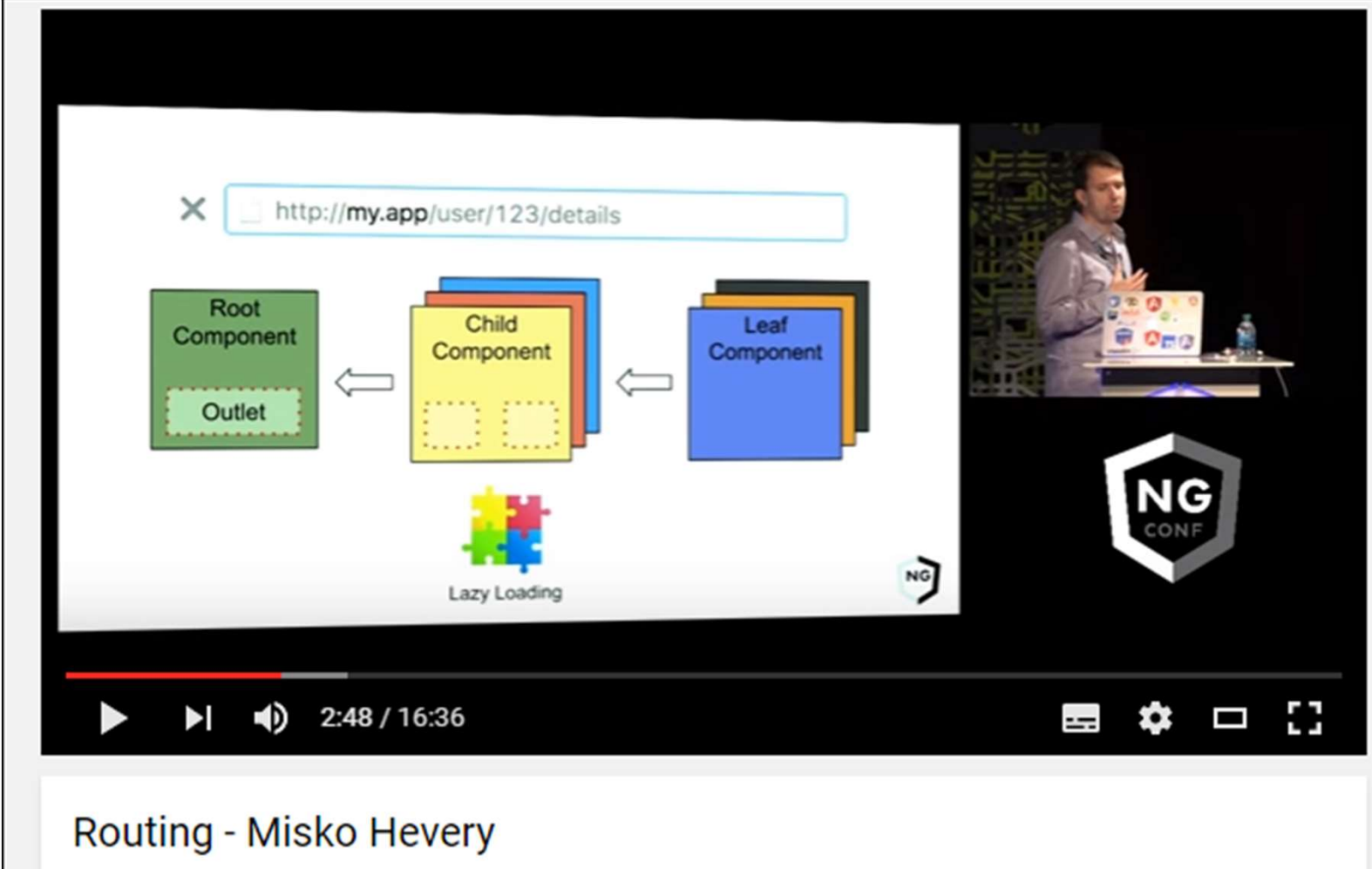
More info

More background information on routing

Meer over routing

- <https://angular.io/docs/ts/latest/guide/router.html>
- <http://blog.thoughttram.io/angular/2016/06/14/routing-in-angular-2-revisited.html>
- <http://blog.thoughttram.io/angular/2016/07/18/guards-in-angular-2.html>
- <https://vsavkin.com/>
- https://angular-2-training-book.rangle.io/handout/routing/child_routes.html

New Component Router



The screenshot displays a video player interface. The main content area shows a diagram illustrating the 'New Component Router' architecture. At the top, a browser address bar displays the URL `http://my.app/user/123/details`. Below this, a diagram shows a hierarchy of components: a green 'Root Component' containing an 'Outlet', which points to a stack of 'Child Component' boxes, which in turn points to a stack of 'Leaf Component' boxes. Arrows indicate the flow of data or control from the leaf components up through the child components to the root. Below the diagram, there is a logo consisting of four colored puzzle pieces (yellow, green, blue, red) with the text 'Lazy Loading' underneath. In the bottom right corner of the diagram area, there is a small 'NG' logo. To the right of the diagram, a small inset video shows a man standing at a podium with a laptop, presenting. Below the main diagram area, a video player control bar shows a progress bar at 2:48 / 16:36, along with play, pause, and volume icons. At the bottom of the video player, the title 'Routing - Misko Hevery' is displayed.

Routing - Misko Hevery

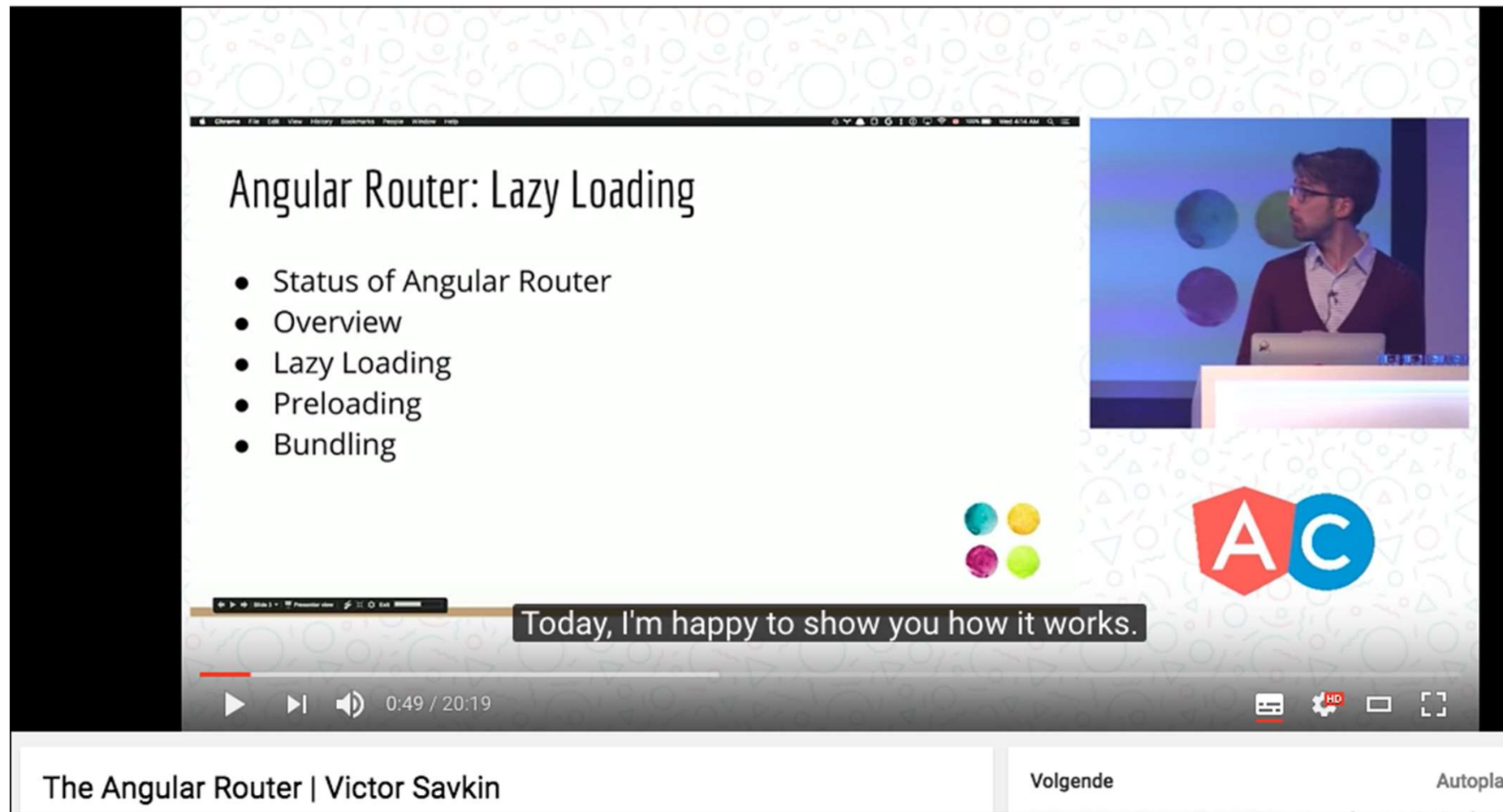
<https://www.youtube.com/watch?v=d8yAdeshpcw>

Victor Savkin (=maker van de router)



<https://leanpub.com/router>

<https://www.youtube.com/watch?v=QLns6s02O48>



The image shows a YouTube video player interface. The video content is a presentation slide titled "Angular Router: Lazy Loading" with a bulleted list of topics: Status of Angular Router, Overview, Lazy Loading, Preloading, and Bundling. The slide also features the Angular logo and a small inset video of the presenter, Victor Savkin. The video player controls at the bottom show the video is at 0:49 of 20:19. The video title "The Angular Router | Victor Savkin" is displayed on the left, and "Volgende" (Next) and "Autoplay" options are on the right.

Angular Router: Lazy Loading

- Status of Angular Router
- Overview
- Lazy Loading
- Preloading
- Bundling

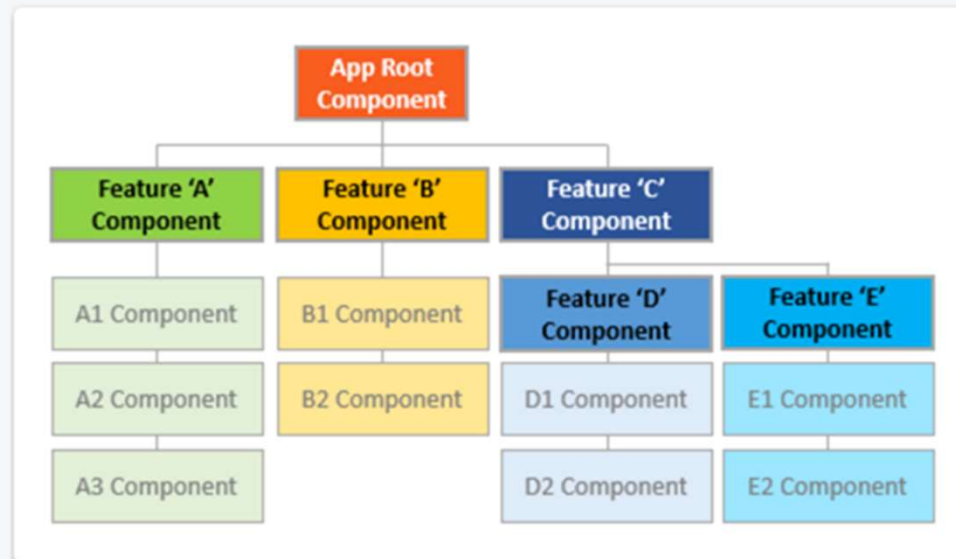
Today, I'm happy to show you how it works.

The Angular Router | Victor Savkin

Volgende Autoplay

Advanced routing

It's looking good as a general pattern for Angular applications.

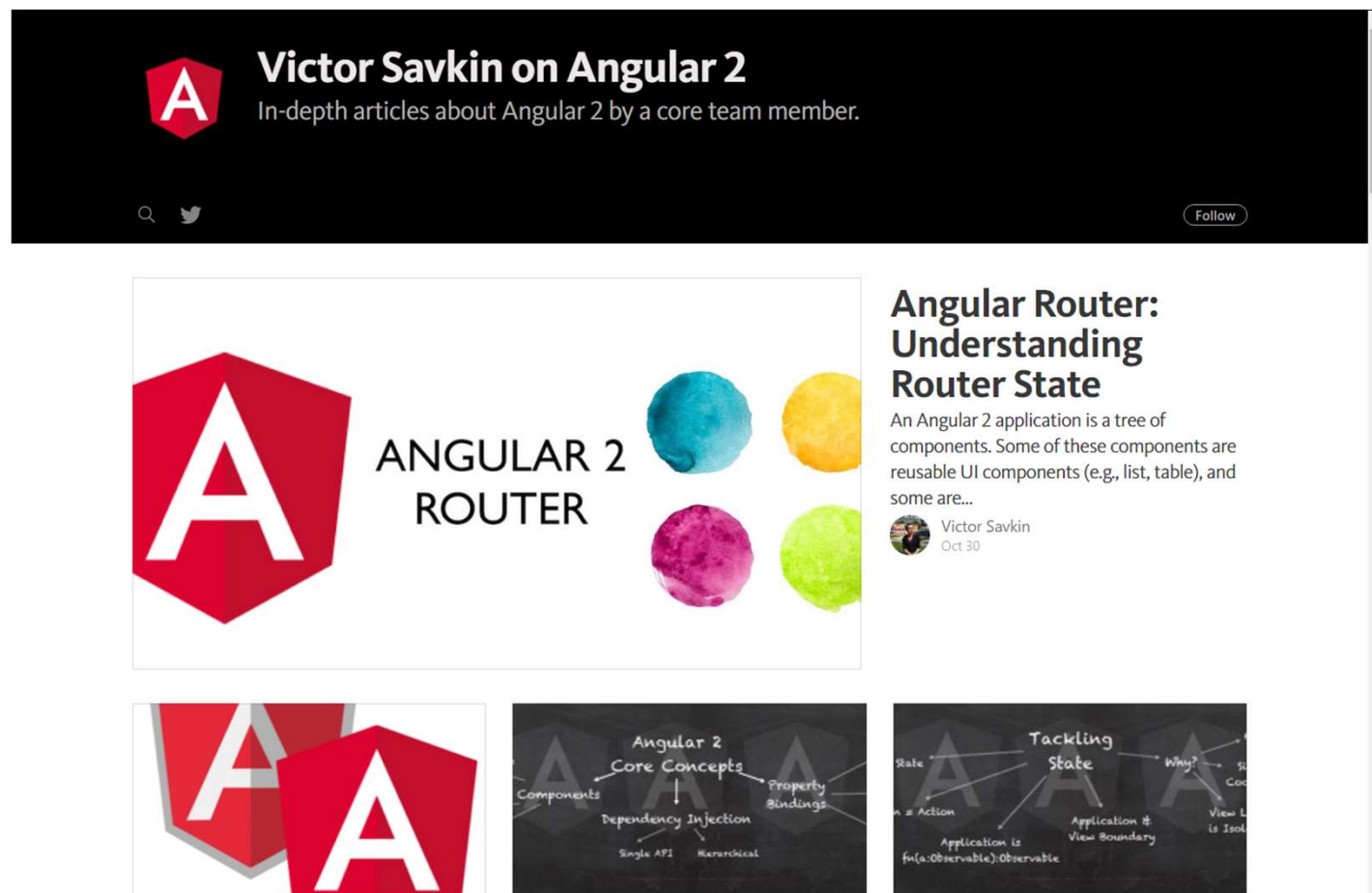


- each feature area in its own module folder
- each area with its own root component
- each area root component with its own router-outlet and child routes
- area routes rarely (if ever) cross

<https://angular.io/docs/ts/latest/guide/router.html>

Victor Savkin on Routing

Victor Savkin – creator of the router



The screenshot shows the header of a website titled "Victor Savkin on Angular 2" with the subtitle "In-depth articles about Angular 2 by a core team member." Below the header is a search bar and a "Follow" button. The main content area features a large article preview for "Angular Router: Understanding Router State" by Victor Savkin, dated Oct 30. The preview includes a large Angular logo and the text "ANGULAR 2 ROUTER". Below the article preview are three smaller images: a stack of Angular logos, a diagram titled "Angular 2 Core Concepts" showing relationships between Components, Dependency Injection, Property Bindings, Single API, and Hierarchical; and a diagram titled "Tackling State" showing relationships between State, Action, Application #, View Boundary, and View L is Isol.

Victor Savkin on Angular 2
In-depth articles about Angular 2 by a core team member.

Angular Router: Understanding Router State

An Angular 2 application is a tree of components. Some of these components are reusable UI components (e.g., list, table), and some are...

Victor Savkin
Oct 30

Angular 2 Core Concepts

- Components
- Dependency Injection
- Property Bindings
- Single API
- Hierarchical

Tackling State

- State
- Action
- Application #
- View Boundary
- View L is Isol

<https://vsavkin.com/>